A Multi Feature Pattern Recognition for P2P-based System Using In-network Associative Memory

Amiza Amir[1 2], Anang Hudaya M.Amin[3], and Asad Khan[1]

[1] Clayton School of IT, Monash University, Melbourne, Australia
{amiza.amir,asad.khan}@monash.edu
[2] School of Communication and Computer Engineering
Universiti Malaysia Perlis, Malaysia
[3] Universiti Teknologi PETRONAS, Malaysia

Abstract. Peer-to-peer (P2P)-based systems encompass powerful capabilities but have been notoriously viewed for copyright infringement issue and recently for sensitive data leaking problem. We aims to provide a pattern recognition capability in solving some problems in P2P network itself(e.g. file pollution, copyright infringement, security issues, etc). As this involves dealing with complex datasets(e.g. image, audio, and video files), a significant effort is required to analyse these datasets thus limits the existing schemes ability to effectively perform recognition. In this paper, we will demonstrate our approach, called Distributed Associative Memory Tree(DASMET) in dealing with multi-feature recognition in P2P-based system. The scheme constructs logical tree like structures within a peer-to-peer network enabling the nodes to search for patterns comprising multiple temporal or spatial features within a fixed number of steps using in-network processing. In doing so, the information held at individual peers is integrated into a common knowledge base, which can be associatively searched by any peer within the network. The scheme incurs low complexity overheads and it is fault tolerant. Finally, we prove the accuracy and scalability of the scheme through our experiment results.

1 Introduction

Current approaches in pattern recognition require significant amount of effort to analyse different forms of features, due to the curse of dimensionality problem. This limits their ability to seamlessly and effectively perform recognition and classification involving complex data sets. Most of the existing schemes incur high computational complexity that inhibits their capability to scale up for increasing number of features. Multiple-feature implementation enables a holistic approach towards pattern recognition procedure that takes into consideration all significant features, which represent a particular set of patterns, such as images and sensor readings. This intends to reduce the bias effect of selecting only a single feature for classification/recognition purposes.

Existing literature on multi-feature recognition has been able to produce high recognition accuracy due to increased number of features to be considered. These include publications in [1, 12, 11, 4]. Most of the work on multi-feature recognition deals with images and optical characters. This is probably due to large number of features that could be extracted from images, including lines, colour, curves, and texture regions. Nevertheless, the complexity of overall recognition scheme also increases due to the large number of features to be analysed. For instance, the multi-feature face detection scheme proposed by [12] implements probabilistic-based AdaBoost to train different features for recognition purposes. This significantly affects the scalability of the scheme due to iterative probabilistic interpretation process to converge these features to global minimum output. In other research, [1] have proposed multiple-expert system and iterative clustering algorithm has been adopted in this implementation. The combination of complex and highly-iterative algorithms makes this approach less scalable. Apart from these studies, [3] have also considering the use of combined classifiers for pattern recognition involving multiple features. In their research, a number of classifiers, including statistical, machine learning and neural networks have been applied for classification process. Nevertheless, the proposed scheme incurs significantly-high computational costs in determining accurate classification output. As a result, the scalability of the scheme deteriorates as the number of training and testing data sets increases.

P2P networks enable direct resource sharing among peers. This makes a system that is built on their platform cost efficient and effective [10]. A node can join and leave a P2P network easily without facing any bureaucracy and some P2P networks can scale up to hundreds of thousands or even to millions of peers. More scalability implies that a larger number of resources can be shared among each other. In this paper, we attempt to design a scalable an efficient(e.g. time and space complexity) pattern recognition scheme by harnessing the resource pooling within the network. It is argued that by having a set of distributed computational networks working together, forming a distributed recognition network will alleviate the issue of scalability of pattern recognition scheme against increasing number of features to be considered.

The paper is organized as follows. In section 2, we briefly review the lightweight distributed pattern recognition scheme for wireless sensor network(WSN), Graph Neuron(GN) and Hierarchical Graph Neuron (HGN) which are the basis of our proposed approach. In section 3, we present our approach for multi feature pattern recognition, its complexity analysis, the required communications and estimates the expected speedup. We briefly discuss the P2P overlay framework that we use in this work in section 4 and explain our mechanism in handling dynamic network in section 5. In section 6, we present the experimental results. Finally, section 7 is the conclusion of our work.

2 Graph Neuron and Hierarchical Graph Neuron

Our proposed approach is adapted from a specially designed distributed pattern recognition scheme for wireless sensor environment called Graph Neuron(GN). Other than its readily distributed nature which makes it easier for us to adapt it in our problem, the strength of this scheme is its simplistic and single cycle of learning.

Graph Neuron is a connected graph with a set of vertices, V and a set of edges, E. The GN network represents all possible points in representing the pattern. Each node, called *neuron*, holds the $\{value, position\}$ pair information, where *value* represents the domain value associated with the neuron and the *position* represents the position of the neuron in the pattern space. Fig. 1 shows the flat GN structure as introduced in [7] of input pattern of size 7 bits, with input values of "1" and "0".

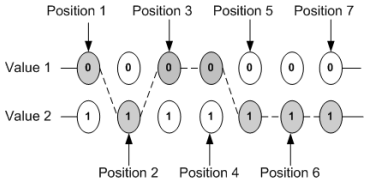


Fig. 1. A flat GN structure for binary pattern with domain value: {0,1} and pattern size seven. The possible values (domain of the pattern) is represented by the number of row while the size of pattern is represented by the number of positions or columns.

The dotted line represents the inter-node communication of the pattern **0100111**. Each neuron communicates with its adjacent nodes: the left and right neighbours. The value corresponding to a specific position in the input pattern are mapped and compared with the neuron at the same position that holds the same value for memorisation (signifies a new input pattern) or recall (signifies an old pattern). However, the flat GN suffers from a crosstalk problem; therefore it cannot be fully relied upon for accuracy. It had been applied in the WSN applications where its fast recall time and high scalability characteristics were proven [6].

Hierarchical Graph Neuron (HGN) [9] is an improvement to the GN algorithm. The crosstalk problem in GN algorithm is eliminated in HGN by constructing a hierarchical structure so that the neuron at the top can perceive the overall pattern (see [9]). However, the number of processing nodes and communication cost are significantly increase in HGN.

In flat GN and HGN schemes, the wireless sensors communicate with a powerful base station, called S&I(Simulator & Interpreter). Compared to the WSN architecture which consists of resource constrained sensors, and a powerful base station; in P2P architecture, there is no server; and all hosts heterogenous. There

is no guarantee to have a single powerful entity which is capable in managing the large amount of information like a base station. On the other hand, peers often have higher processing capability and larger storage than the wireless sensors. Therefore, HGN is not suitable to be applied in the P2P networks. Moreover, the proximity of peers in P2P networks with logical topology does not guarantee the physical proximity between peers. Therefore, the communication overhead should be minimal to avoid any delay caused by the network.

3 Distributed Associative Memory Tree(DASMET)

Our approach is viewed as a multi sensory system which each peer in a P2P system acts as a sensor. Each sensor is responsible to response to a small subset of features in a feature space. Large dimensional feature vectors are divided into small segments and these segments are learnt by several different peers separately. The results are then combine iteratively in a tree structure until converge at the root. Therefore, it is scalable in terms of handling a high dimensional multi-feature dataset. In DASMET, the responsibility to interpret the result, which is usually handled by a single base station, is handled through a collaboration of peers. DASMET is an associative memory algorithm to facilitate an accurate, efficient and scalable pattern classification in a P2P-based distributed environment. DASMET inherits the single cycle learning and accuracy characteristics of HGN but significantly reduces the number of nodes and the communications required in HGN. The trade-off is that a node in DASMET have to undertake a greater workload than a node in an HGN but the magnitude of this workload is reasonable for a peer in a P2P file sharing system(usually a common user's PC).

3.1 Structure

The structure of a DASMET is a rooted tree $G = (V, E)$, where V is the set of vertices and E is the set of edges (see Fig. 2). Each vertex in DASMET is an entity or a computational unit, x, that is $x \in V$. The structure depends on two restrictions: a) the maximum number of children of each node, φ. This is equal to the maximum input connection(in-degree) allowed of a node b) the target (maximum) sub-pattern size a leaf node can handle, ϖ. Given a pattern with dimension ρ, the number of leaves, n_H is equal to $\lceil \frac{\rho}{\varpi} \rceil$. We classify the entities in DASMET into three roles: a) rootnode, R - the root node which is responsible to combine and finalise the result; b) leadernode, S - all internal nodes which are responsible to collect and combine the result from nodes at lower level; and then propagate the results to the nodes at upper level; and c) hubnode, H - all leaf nodes which are the nodes without children. These node are responsible to recognize or classify a segment of pattern locally and independently.

A pattern feature vector, P is fractioned into several k small segments and each of the segment is processed by a hubnode. For position ith of the sequence of segments, a segment, p_i is assigned to a hubnode, h_i. Note that the term

position in this context means the order of a feature or a feature segment within the feature vector. For instance, a binary image can be viewed as a sequence of a binary string where each pixel $Pix\{val, loc\}$ is the combination of its binary value val and spatial loc information. Each hubnode is responsible for local training and the recall of a segment of pattern. The results are successively calculated by the nodes in the upper layer until the final result is calculated at the rootnode. When dealing with a large pattern size, we have a greater number of segments, and thus the parent peer that is responsible to aggregate the result will be burdened with too many received transactions to process. The computational load amongst the nodes is maintained by growing the DASMET tree. This increases the scalability of the algorithm in handling high number of features. Vertices are added with their edges connecting to the parent node and the workload is distributed among these nodes. The process of adding vertices and dividing the workload is executed successively until a reasonable amount of workload (defined by ϖ) to be handled by a leaf node is achieved. Fig. 2 shows an example of a DASMET structure for a binary pattern size-40. This pattern is partitioned into 8 segments with segment size-5 each.

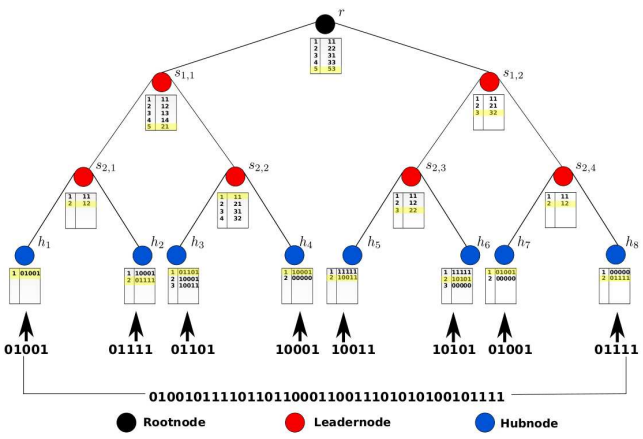


Fig. 2. Example of DASMET architecture for a binary pattern with size 40. The predefined maximum children of each node, φ is 2 and ϖ is 5. The input pattern is segmented into 8 segments and each segment is assigned to a hubnode respectively. It has 8 hubnodes and 6 leadernodes with the height of the tree is 3. Each node holds a *biasArray*(represented by the box) which stores the memory of the learnt pattern. The yellow coloured entries represent the active entries for the input pattern.

Given the length of pattern ρ and the number of nodes at level i, n_i is at most φ^i. The height of an DASMET tree $\hbar$ is $\hbar \geq log_{\varphi} \left\lceil \frac{\rho}{\varpi} \right\rceil$. The total number

of nodes in DASMET, $|V| = N$ is:

$$N \leq \sum_{l=0}^{\hbar-1} \varphi^l + \left\lceil \frac{\rho}{\varpi} \right\rceil \tag{1}$$

The total number of edges in DASMET, $|E|$ is equal to $N - 1$.

Local Storage Each node holds a table called *biasArray* with each entry in the table represents the memory of a pattern that has been occured to each node. The format of a *biasArray* is vary to the node's role. An entry of a *biasArray* for a hubnode is the value of pattern's segment. While for a leadernode or rootnode, the entry for each column in a *biasArray* consist of a pair of $\langle$*biasIndex*, *position*$\rangle$ from its child. *position* is the position of a node from the parent's view and the information associated with the *position* termed as *biasIndex*. A *biasIndex* for column j in a *biasArray* of a leadernode or a rootnode is the index of the active entry from its child at position jth. Therefore the number of column in a *biasArray* of a leadernode or a rootnode is equal to number of children it has.

Learning Procedure The nodes in DASMET perform a basic recall and learning procedure similar to the GN model[7]. *Coordinator* segments the training pattern, P into a sequence of several small sub-patterns $\{p_1, p_2, \ldots p_j\}$ such that p_i is a sub-pattern at ith position in P and then sends these sub-patterns to the hubnodes.

- *Hubnode.* A hubnode receives input sub-pattern from it *Coordinator*. Then, the lookup for the sub-pattern is performed through the *biasArray* and if it is not found, then the sub-pattern is stored as a new entry. Therefore, the resulting expression for learning at hubnode is $0(\frac{\rho}{n_H} \cdot n)$.
- *Leadernode and Rootnode.* A leadernode receives k reports from its children (formed as *biasEntry*) where k is equal to the number of children it has and $k \leq \varphi$. The learning and recall procedure in leadernode involves comparing indices in *maxEntries* with the pertaining element in *biasArray* for each report from its child, which is $a_{(b,i)}$, for $i = 1, 2, \ldots, n$; and $b = 1, 2, \ldots, k$. The learning time taken, at each leadernode or rootnode is $O(k \cdot n)$.

Recall Procedure The recall is executed similarly with the learning except there is no new entry is inserted into the *biasArray*. Each node in a DASMET creates a temporary structure called a *voteTable*, which hold the similarity (vote score) of the pattern to recall with the stored patterns in *biasArray*. The vote is calculated by obtaining the frequency of matched element in each column j in *biasEntry* with corresponding column j in each row i in *biasArray*. Each row represents a stored memory. A lookup element in a hubnode is the value of a pattern at position j while a lookup element in a leadernode/rootnode is a vector of *maxEntries* that is received from its child at position j. *maxEntries* is the indices of the entries with the highest vote at a child node. The element

is searched through the column j in *biasArray* until a match is found. As it is possible to get more than one matched index, the votes for every matched indices are incremented by one.

- *Hubnode.* In recall procedure, *voteTable* is updated simultaneously with the maximum vote and *maxEntries* list. This is performed each time the similar value at each position in the input sub-pattern with each column b in the *biasArray* is found. The total number of column is $\frac{\rho}{n_H}$. Thus the recall execution in each hubnode has time complexity $O(\frac{\rho}{n_H} \cdot n)$.
- *Leadernode and Rootnode.* A leadernode receives k reports from its children. The recall procedure in leadernode involves comparing indices in *maxEntries* with the pertaining element in *biasArray* for each report from its child, which is $a_{(b,i)}$, for $i = 1, 2, \ldots, n$; and $b = 1, 2, \ldots, k$. The calculation to update *voteTable*, maximum vote and *maxEntries*,(we termed as $f(rep)$ from now on) is performed at each time a report is received by a node from its child. Given q is the size of *maxEntries*, the time required to calculate $f(rep)$ is $O(q \cdot n)$. Considering k reports are received from k children, the recall time taken at each leadernode or rootnode is $O(k \cdot q \cdot n)$.

However, the product of $k \cdot q$ is much smaller than n. Therefore the $k \cdot q$ is negligible compared to n. Hence, the complexity of $f(R_s)$ is bounded by n and we may conclude that the time complexity for both operations, learning and recall at a leadernode is linear in time. While the above complexity analysis is based on linear search(unordered data), the complexity of DASMET could be improved to $O(\log n)$(binary search).

3.2 Communication Cost

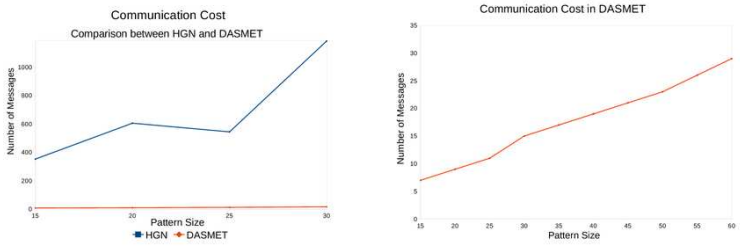
The total number of communications during learning and recall in DASMET scheme is the total number of edges in the DASMET structure and with additional communications needed when DASMET communicates with the external entities(e.g: coordinator nodes).

$$M_{DASMET} = (N - 1) + n_H + 1 \tag{2}$$

We plotted the number of messages that is required in HGN and DASMET as shown as in the Fig.3.2. The bigger size of a pattern results to the dramatically increase of messages in HGN. However, the number of messages in DASMET remains low and increases very slowly.

3.3 Speedup

As the algorithm works in parallel, the complexity of the learning/recall process is the sum of the processing involves from the farest leaf node to the rootnode. This including the communication between a *Coordinator* and a hubnode during a pattern input into DASMET; the communication within DASMET network (where height of the tree $\hbar$ is equal to the number of required edges(messages)



(a) Comparison of communication cost in HGN and DASMET

(b) Communication cost in DASMET

Fig. 3. The DASMET structure in this example is defined by $\varphi = 5$ and $\varpi = 5$. Fig. 3(a) shows that the number of messages that is required in HGN increases significantly compared to DASMET with the increase of pattern size. Fig. 3(b) shows the small increase of communication cost in DASMET.

from a hubnode to a rootnode); and the communication between rootnode and *Coordinator* when the rootnode reports its result to *Coordinator*. The number of leadernodes in this calculation is $(\hbar - 1)$ and given communication time to send each message is T_{comm}. Given $k \leq \frac{\rho}{n_H}$ and the serial implementation of the algorithm cost $O(\rho \cdot n)$. The estimate speedup of our learning scheme when dealing with high number of features ρ:

$$Sp_L = \frac{O(\rho \cdot n)}{(\hbar + 1)(O(\frac{\rho}{n_H} \cdot n)) + (\hbar + 2)T_{comm}}$$

3.4 Multi DASMET Network Approach for Large Training Datasets

In a P2P networks, there could be hundred thousands to millions of nodes join a network at time. Therefore, it is possible to have several DASMET networks within a P2P network. To increase the scalability for large training dataset, the training set is divided into several buckets and each bucket is assigned to a DASMET as shown in Fig. 4.

In this case, the indices with the highest votes from each subnet are sent to the combiner and the index with highest votes amongst these highest votes is selected as the winner.

4 P2P Overlay Platform: Kademlia

In this research, Kademlia [8] is used as a P2P platform for our scheme. In Kademlia, stale nodes or nodes which are often connected for a short time are removed from the routing table thus provides us a mechanism to build a strong network with less unstable peers on the network. The provided uptime information is useful so that we can assign main jobs to the more stable peers. Kademlia uses tree-based routing and a prefix-based routing table, similar to Pastry. In

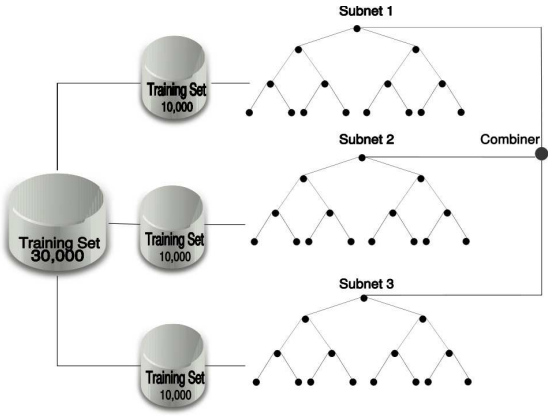


Fig. 4. An example of a multi DASMET structure which consist of three DASMET networks work collaboratively to enhance the system's scalability. A large training dataset with size-30, 000 is partitioned into three buckets with 10, 000 training data in each bucket. Each bucket is trained by a DASMET subnet. The results from all subnets then compiled by the a combiner node which responsible to calculated the final result.

contrasts to the Pastrys recursive routing, Kademlia uses iterative routing, where a peer contacts peers with incrementally smaller XOR distances to the lookup key. It uses XOR distance metric where XOR of two node's ID is use to determine the distance between them. Kademlia is asymmetric where the distance between node b and a is $d(a, b) = d(b, a)$. The latency is logarithmically increasing with an increase number of peers which shows the scalability of Kademlia.

5 Handling Dynamic Network

Abrupt peers leaving the network effects the availability and prohibits the system from working correctly. The storage entity in our PR scheme called *BiasArray* is a dynamic entity with frequent data updates similar to a database but in much lighter version. Therefore, we implement a consistent and secure backups update using byzantine fault tolerance replication protocol [2] to ensure data consistency and freshness at all peers. We use replication method for data redundancy across the network to ensure data availability and scheduled maintenance mechanism to maintain the availability of data.

Data Replication We ensure the reliability and availability of GN nodes by replicating the information at several peers. By creating r replications of a GN node on r peers, we could reduce the possibility of losing bias array information up to r factor. A peer may responsible for a particular session before failure or go offline and another peer will be able to replace it without a significant degradation to the system. Each GN node has a backup with is called *Shadow*

and r replications of itself. We termed the original node as *Source* and it has a backup node called *Shadow* which responsible to take over the *Source* role whenever *Source* fail. Each *Source* has another backups called replica. In order to synchronize local replicas, a consistent communication and control mechanism is provided through byzantine fault tolerance replication protocol [2]. The *Source* and its backups are placed at different peers. Replicas have the same state with one another and correctly clone the *Source* GN node.

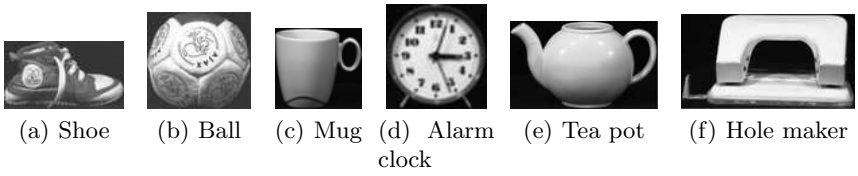
Periodically Recovery Mechanism PR network is periodically recovered through a scheduled recovery mechanism within a recovery interval RI. All nodes in PR network works collaboratively to detect any change to the system and perform a node's recovery if needed. All children nodes act as failure detector of the parent node. Each *Source Peer* except the rootnode's *Source Peer* periodically contacts its parent to ensure that it is alive. For hubnode's *Source Peer*, its parent detects the failure by ensuring all children(hubnode's *Source Peer*s) constantly communicate with it. The hubnode's *Source Peer*s does not contact the parent within certain duration, the parent contacts the the hubnode to check if the node is still alive. All detected failure is reported to the *Shadow Peer*. When a *Shadow Peer* receives a failure report, it contacts the *Source Peer*. If there is no response from *Source Peer*, *Shadow Peer* takes over the *Source Peer* role. The node selection algorithm then performed to select the suitable candidate to replace itself for a *Shadow Peer* position. *Shadow Peer* might also go offline. In that case, the replicas of the GN node can be found at the close neighbours of *Source Peer*.

6 Experimental Result

To test the accuracy of DASMET in dealing with multi-feature datasets, we used ten training images [5] (as shown in Fig. 5). The collection we used in this experiment is obtained from The Amsterdam library of object images [5]. In preprocessing, these images were resampled into 30×50 and then converted into binary format. Thus, the number of features for this datasets is 1500.

For each of these images, we created 1000 versions by adding random distortion to the original image. Started from the first 1000 versions with 10% random distortion, we generated the next 1000 versions with 15% distortion and then slowly increase the distortion rate up to 50%. Then we compiled the distorted versions into five testing datasets(2000, 4000, 6000, 8000, and $10,000$) where 10% from the dataset are the distorted versions of each image. Therefore, in total we have 45 dataset those are used in this experiment In this experiment, $\varpi = 10$ and $\varphi = 5$. Hence, we have a DASMET structure which has 231 nodes and 4-height. Each learning and testing cycle incurred 380 messages amongst nodes.

Fig. 6 demonstrates the accuracy of our distributed algorithm. We could conclude that the error rate in DASMET is low with only 5% possibility of incorrect prediction when given a pattern with a very high distortion which up



(a) Shoe (b) Ball (c) Mug (d) Alarm clock (e) Tea pot (f) Hole maker

(g) Teddy bear (h) Book (i) Towel (j) Hat

Fig. 5. Ten raw images for accuracy testing labelled fig. 5(a) Shoe, fig. 5(b) Ball, fig. 5(c) Mug, fig. 5(d) Alarm clock, fig. 5(e) Tea pot, fig. 5(f) Hole maker, fig. 5(g) Teddy bear, fig. 5(h) Book, fig. 5(i) Towel, and fig. 5(j) Hat.

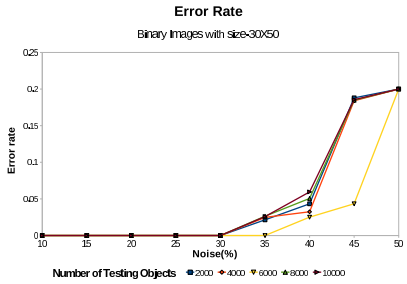


Fig. 6. Figure shows the error rate in 30×50 binary images recognition by using DASMET. The error rate is remain low with the increase of distortion rate.

to 35% noise. The error rate starts to increase significantly from 40% distortion rate for all testing datasets. We obtained a consistent result when tested the algorithm with different size of testing dataset(from 2000 to 10000). It shows that the size of testing dataset does not affect the accuracy. The result shows that the scheme is highly accurate and also show the scalability of the scheme in handling multi feature datasets.

7 Conclusion

In this paper, we introduce and analyse a variant of GN algoritm for multi feature pattern recognition within P2P network. The number of processing nodes and communications are significantly improved in our approach compared to the HGN algorithm. The scalability is achieved through distributing and linking a massive number of training objects/memories over a P2P network which may

involves hundred thousands to millions of nodes. By breaking the dimension of pattern into smaller parts, computing these parts separately in parallel and then combining the recognition result iteratively in a tree structure, we significantly reduce the computation complexity at a node in terms of both, space and computation time. Space complexity at each peer remains low as several different patterns might have multiple repeating same sub-patterns and thus a repeating sub-pattern is only stored once (at the first time it occurs). This approach uses a single cycle learning as objects in a P2P environment are massive and frequently updated. In future work, we intend to investigate the possibility of using this structure to parallel the available well-established schemes(e.g. naive bayes).

References

[1] Cao, J., Ahmadi, M., Shridhar, M.: Recognition of handwritten numerals with multiple feature and multistage classifier. Pattern Recognition 28(2), 153–160 (1995)

[2] Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems (TOCS) 20(4), 398–461 (Nov 2002)

[3] Duin, R.P.W., Tax, D.M.J.: Experiments with classifier combining rules. In: MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems. pp. 16–29. Springer-Verlag, London, UK (2000)

[4] Hongtao, S., Feng, D.D., Rong-chun, Z.: Face recognition using multi-feature and radial basis function network. In: VIP '02: Selected papers from the 2002 Pan-Sydney workshop on Visualisation. pp. 51–57. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2002)

[5] Jan-Mark Geusebroek, G.J.B., Smeulders, A.W.M.: The amsterdam library of object images. International Journal of Computer Vision 61(1), 103–122 (2005)

[6] Khan, A.I., Baqer, M., Baig, Z.A.: Implementing a graph neuron array for pattern recognition within unstructured wireless sensor networks. Lecture Notes in Computer Science pp. 208–217 (2006)

[7] Khan, A.I.: A peer-to-peer associative memory for intelligent information systems. In: Proceeding Thirteenth Australasian Conference on Information Systems. vol. 1 (2002)

[8] Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. pp. 53–65. IPTPS '01, Springer-Verlag, London, UK (2002), http://portal.acm.org/citation.cfm?id=646334.687801

[9] Nasution, B., Khan, A.I.: A hierarchical graph neuron scheme for real-time pattern recognition. IEEE Transactions on Neural Networks pp. 212–229 (2008)

[10] Sahin, O.D., Emekci, F., Agrawal, D., Abbadi, A.E.: Content-based similarity search over peer-to-peer systems. In: Proceedings of DBISP2P04. pp. 61–78 (2004)

[11] Yu, D., Ma, L., Lu, H.: Lottery digit recognition based on multi-features. In: Systems and Information Engineering Design Symposium, 2007. SIEDS 2007. IEEE. pp. 1–4 (2007)

[12] Zhu, H., Zhang, L., Sun, H., Xiao, R.: Face detection using multi-feature. In: Advances in Cognitive Neurodynamics ICCN 2007, pp. 921–925. Springer Netherlands (2008)